Nathaniel Serrano

COS 420

Program 5

3.2.2025

Lab Book

Time spent working with AI: March 2, 9:00 pm - 9:45 pm

Get rid of "104"

This section consisted of creating a constant that contains the winning score value. Fortunately, I already had this feature implemented in the previous assignment after hearing it get talked about in class. I simply requested the feature in my previous prompting session with ChatGPT 4o by writing, "replace all occurrences of 104 with a new constant called WINNING_SCORE set to that value." This was a simple task for the AI tool and was completed without error.

Consolidate Randomness

The Consolidate Randomness section required the creation of a new Dice class where all random rolls are handled. This prompt was simple, since the assignment instructions already worded the request in a clear manner. My prompt began by giving the AI tool the Prog6.java file from Program 2. Next, I simply copied what was written in the assignment instructions: "Create a new class Dice that contains only one random number generator… Player objects can create their own Dice objects as needed; making them instance variables is probably efficient." The AI tool's output was found to end up working perfectly after running a few test games. It is important to note that the Dice implementation also required some changes to the Player.java class. I originally only presented the AI tool with Prog6.java, so it created its own Player class within that file to add the Dice implementation. I simply took the two newly created lines of code and pasted them into the original Player.java class, making sure to update the documentation to be up to date with this change.

Javadoc

This section required the addition of documentation in the style of Javadoc. A similar task was conducted in Program 4 but I decided to change my approach this time around. Previously, I simply requested the AI tool to "document the code in the style of Javadoc." This produced adequate documentation but it looked notably different from my own documentation in the Javadoc style. Aspects like spacing, stating the name of the class at the top, and incorporating the author and date the code was most recently edited were found in my own documentation but not the AI tool's. As such, this time around I included Prog6.java from Program 1 (written by me) for the AI tool to use as reference when writing the documentation. I found this worked wonderfully, with me only having to edit the date to that of today's as well as request ChatGPT 4o to add itself to the list of authors.

<u>File Structure</u>

Lastly, this section required the reorganization of classes into separate Java files. As was described in the instructions, all the code outputted by the AI tool was combined into one long Java file, making it a bit of a hassle to scroll through to find what you are looking for. To rectify this, I first wrote, "Lastly, each class in this file should be its own separate file. Reorganize the classes into separate files while keeping the documentation you just implemented." This led to an interesting result due to the nature of ChatGPT 4o. One big difference between ChatGPT 4o and its predecessors is that code now appears in a much larger window of the screen by default. Unfortunately, this window is only capable of displaying one "file" at a time, and as the AI tool wrote each class in its own file, the contents in the window were simply being overwritten. To correct this, I requested, "Can you have each class in its own code block for me to copy and paste?" In response, the AI tool outputted everything in distinct code blocks as it would in its previous versions, like ChatGPT 3.5. This scenario showcased an interesting downside to using a more modern version of the tool. Once this was done, I simply created the new classes in Eclipse and copy-pasted everything into its respective location, which ended up working with zero issues.

<u>Conclusion</u>

This assignment certainly had a smaller workload compared to previous ones. The goal of it was to determine how AI tools can help with refactoring; whether they accidentally create bugs when rearranging things. In this respect, ChatGPT 4o performed impressively, since it never actually introduced any bugs. All testing done when implementing the new code showed the program worked as it previously did. The minor hiccup I experienced in the File Structure section actually made me glad I chose one of the most recent versions of ChatGPT, since that issue would not have occurred in earlier versions. It shows that while the AI tools are constantly evolving and becoming better at what they do, they are by no means perfect, in both what they output and how they display it.